Luminosity Information for EBs

from the Luminosity Group

There are two ways that users calculate luminosity. Some use getLuminosity directly (see instructions at $www - d0.fnal.gov/d0dist/dist/pagckages/lm\_tools/devel/doc$) and some use the script lumitool.py, which is supported mainly by Michele Weber (see the D0Wiki top group page "How to Lumi" for instructions on using lumitool.py). lumitool.py is a wrapper around data quality and getLuminosity, and it helps the user prepare all the files and commands used by genLBNtables and getLuminosity. Either method should work, and hopefully the instructions are clear for both.

Two flags in getLuminosity can make a significant difference in the calculated luminosity. They are –ignoremissingstreams and –ignoreduplicates. The script lumitool.py also accomodates these options.

If the analyser is sure that he/she is removing duplicate events in their analysis, it is safe to use the –ignoreduplicates flag in getLuminosity. If this flag is not set, getLuminosity will label as bad all LBNs which appear more than once, which protects the user from duplicate events but may result in loss of luminosity.

With the –ignoremissingstreams flag set, getLuminosity makes a correction for any missing streams in the data (assuming data is equally spread over all four streams). If this flag is not set, getLuminosity will declare bad any LBNs which are missing one or more streams.

Datasets should not have duplicate LBNs, and they should not be missing data streams, but both have happened in the past, resulting in significant loss of luminosity for some analyses. Not using these flags will NOT lead to incorrect results, just a loss of luminosity.

New version of dq_defs are released several times a year. The most recent version should always be used–it is the most correct and the most up-to-date. The data quality produces a list of bad runs and bad LBNs, and getLuminosity produces its own list of bad LBNs. These lists should be merged, and then both the bad LBN and bad runs list should be folded back into the analysis so that bad data is not included in the final results. The bad runs and bad LBN lists should also be applied to Monte Carlo files, since it is possible that Monte Carlo was generated with zero bias overlays from runs or LBNs that the user is labeling as bad.

The following questions should be asked during the review of every analysis that relies on a luminosity determination:

1) Are you using a single trigger in your analysis or are you OR-ing multiple triggers together?

If using a single trigger:
2) Did you use that same trigger when using the tools provided by the luminosity group ($www - d0.fnal.gov/d0dist/dist/pagckages/lm\_tools/devel/doc$) to determine the integrated luminosity for your analysis?

If using an OR of triggers:
2) How did you determine the integrated luminosity for your analysis? Which triggers did you use? Are any of the triggers in your OR prescaled, and if so how did you take the prescales into account, through the luminosity normalization or the trigger efficiency? Did you discuss your approach with the trigger studies group? See D05329 for a discussion of OR-ing triggers.

3) Did you use the most recent version of data quality definitions (dq_defs)? Did you use the identical definition of data quality when determining the trigger efficiency, calculating the luminosity, and in the analysis of the dataset? Did you merge the bad LBN list from data quality with the bad LBN list produced by getLuminosity and fold that merged list back into your analysis? Did you also apply the bad run and bad LBN lists to your Monte Carlo files?

4) Did you correct either the luminosity or your efficiency for the CAL event quality flags? Note that the zero bias sample used for the CAL event quality flags must have the same list of good or bad LBNs as your analysis. This correction is of order 3%.

5) How are you handling possible duplicate events in your dataset? Are you using the –ignoreduplicates flag in getLuminosity? If so, how are you removing duplicate events in your analysis?

6) How are you handing possible missing streams in your dataset? Are you using the –ignoremissingstreams flag in getLuminosity?